Due Date: November 18, 2008

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**
**BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES**

| | | |
|---|---|---|
| In re Application of: | ) | |
| | ) | |
| Inventors: John G. Beltran et al. | ) | Examiner: Omar R. Abdul-Ali |
| | ) | |
| Serial #: 10/781,307 | ) | Group Art Unit: 2178 |
| | ) | |
| Filed: February 18, 2004 | ) | Appeal No.: _____ |
| | ) | |
| Title: DYNAMIC PROPERTIES FOR SOFTWARE OBJECTS | ) | |

**REPLY BRIEF OF APPELLANTS**

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Sir:

In accordance with 37 CFR §41.37, Appellants hereby submit the Appellants' Reply Brief on Appeal in response to the Examiner's Answer dated September 18, 2008 in view of the final rejection in the above-identified application, as set forth in the final Office Action dated January 24, 2008 and the Advisory Action dated April 10, 2008.

No fee is due at this time. However, please charge any additional fees or credit any overpayments to Deposit Account No. 50-0494 of Gates & Cooper LLP.

## I. REAL PARTY IN INTEREST

The real party in interest is Autodesk, Inc., the assignee of the present application.

## II. RELATED APPEALS AND INTERFERENCES

There are no related appeals or interferences for the above-referenced patent application.

## III. STATUS OF CLAIMS

Claims 1-18 remain in the application.

Claims 1-18 stand rejected.

The rejections of claims 1-18 are being appealed.

## IV. STATUS OF AMENDMENTS

Subsequent to the final rejection, no claims have been cancelled, amended, or added.

Subsequent to the final rejection, the specification was amended to overcome objections related thereto (i.e., capitalizing the use of a trademark). Appellants assume that such amendments were entered into the record. However, no indication has been received indicating whether the amendments were officially entered into the record.

## V. SUMMARY OF CLAIMED SUBJECT MATTER

There are essentially two sets of claims. The first set includes independent claims 1, 5, and 17. The second set includes claims 9 and 13. Each set will be addressed separately herein.

### Claims 1, 5, and 17

These independent claims provide the ability to create and display a dynamic property on a per object basis. As explicitly set forth in the claims, the dynamic property is created at runtime for an object instance on a per-instance basis and is not stored with the object. Referring to claim 1, the first limitation provides for retrieving a reference to an object instance. This element provides an instance of an object/class for which the dynamic property will be created.

The second limitation retrieves a reference to a property source instance. Appellants note that the limitation provides that it is an instance of the property source (i.e., it is an instance of an object). Further the reference to this property source instance is obtained. The second limitation

also provides that the reference is retrieved from "an association between the object and the property source instance". Thus, rather than retrieving a reference from a random location, it is retrieved from an entity identified as an "association" in the claims. Such an association can take a variety of forms (e.g., a mapping, a linked list, etc.). In addition, the second element provides that the property source instance creates and supplies the dynamic property and an initial value for the dynamic property for/to the object instance. Accordingly, when examining this claim limitation in combination with the first limitation, the claims explicitly provide that a property source instance creates (at run time) and supplies (at run time) a new property and an initial value for the new property to/for the object instance. Thus, rather than the object instance creating a value for one of its own properties at runtime (e.g., a name for the object instance), this claim element explicitly provides that a property itself AND a value for that property is created for one object (i.e., the object instance) by another object (i.e., the property source instance).

The third limitation provides the two references (i.e., the reference to the object instance and the reference to the property source instance) to a control. The third limitation further provides that the control is configured to retrieve the dynamic property from the property source instance and display the dynamic property in a user interface.

As can be seen by the above limitations, there are specific and detailed claim limitations that provide a precise structure and capability. More specifically, a property source instance creates a property (and a value for that property) for an object instance. Such a creation is performed dynamically during runtime.

| CLAIM LIMITATION | SPECIFICATION SUPPORT |
|---|---|
| 1. A computer-implemented method for displaying per-instance dynamic properties of an object comprising: | [0005]-[0006] - Page 2, line 23-page 3, line 20; [0010] - Page 4, line 22-page 5, line 7; [0034] - page 11, lines 4-14; Fig. 4. |
| (a) receiving a reference to an object instance having a dynamic property that | [0010] - Page 4, line 22-page 5, line 7; [0034]-[0035] - Page 11, lines 5-21; FIG. 4 - 402-404; |

| | |
|---|---|
| is created at runtime for the object instance on a per-instance basis and is not stored with the object; | [0045] Page 15, lines 4-12; [0048]-[0049] - Page 16, line 13-page 17, line 8; FIG. 5 - 502; [0053] - Page 18, lines 3-12; |
| (b) retrieving a reference to a property source instance from an association between the object and the property source instance, wherein the property source instance creates and supplies the dynamic property and an initial value for the dynamic property for/to the object instance; and | [0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504; |
| (c) providing the reference to the object instance and the reference to the property source instance to a control, wherein the control is configured to: | [0003] - Page 2, lines 12-15; [0039] - Page 12, line 22-page 13, line 8; [0042] - Page 13, line 20-page 14, line 3; [0051]-[0053] - page 17, line 14-page 18, line 12; FIG. 5, steps 508-516; [0054]-[0058] - Page 18, line 15-page 20, line 3; FIG. 4, item 410; [0062] Page 21, lines 8-12; |
| (i) retrieve the dynamic property from the property source instance; and | [0045] - Page 15, lines 4-12; FIG. 4, item 410. |
| (ii) display the dynamic property in a user interface. | [0045] - Page 15, lines 4-12; FIG. 4, item 410. |
| | |
| 5. A computer-implemented system for displaying per-instance dynamic properties of an object comprising: | [0005]-[0006] - Page 2, line 23-page 3, line 20; [0010] - Page 4, line 22-page 5, line 7; [0034] - page 11, lines 4-14; Fig. 4. |
| (a) a computer; | [0022] - Page 7, lines 9-17; FIG. 1, 100. |
| (b) an application executing on the | [0023] - Page 7, line 18-page 8, line 7; FIG. 1, |

| | |
|---|---|
| computer; | 108; |
| (c) an object instance, in the application, having a dynamic property that is created at runtime for the object instance on a per-instance bases and is not stored with the object; | [0010] - Page 4, line 22-page 5, line 7; [0034]-[0035] - Page 11, lines 5-21; FIG. 4 - 402-404; [0045] Page 15, lines 4-12; [0048]-[0049] - Page 16, line 13-page 17, line 8; FIG. 5 - 502; [0053] - Page 18, lines 3-12; |
| (d) a property source instance, in the application, wherein the property source instance creates and supplies the dynamic property and an initial value for the dynamic property for/to the object instance; | [0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504; |
| (e) an association, in the application, between the object and the property source instance; and | [0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504; |
| (f) a host, in the application, configured to: | [0037] - Page 12, lines 7-16; FIG. 4, 412. |
| (i) retrieve a reference to the object instance; | [0010] - Page 4, line 22-page 5, line 7; [0034]-[0035] - Page 11, lines 5-21; FIG. 4 - 402-404; [0045] Page 15, lines 4-12; [0048]-[0049] - Page 16, line 13-page 17, line 8; FIG. 5 - 502; [0053] - Page 18, lines 3-12; |
| (ii) retrieve a reference to the property source instance from the association; and | [0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504; |

| | |
|---|---|
| (iii) provide the reference to the object instance and the reference to the property source instance to a control, wherein the control is configured to: | [0003] - Page 2, lines 12-15; [0039] - Page 12, line 22-page 13, line 8; [0042] - Page 13, line 20-page 14, line 3; [0051]-[0053] - page 17, line 14-page 18, line 12; FIG. 5, steps 508-516; [0054]-[0058] - Page 18, line 15-page 20, line 3; FIG. 4, item 410; [0062] Page 21, lines 8-12; |
| (1) retrieve the dynamic property from the property source instance; and | [0045] - Page 15, lines 4-12; FIG. 4, item 410. |
| (2) display the dynamic property in a user interface. | [0045] - Page 15, lines 4-12; FIG. 4, item 410. |
| | |
| 17. A computer-implemented system for displaying per-instance dynamic properties of an object comprising: | [0005]-[0006] - Page 2, line 23-page 3, line 20; [0010] - Page 4, line 22-page 5, line 7; [0034] - page 11, lines 4-14; Fig. 4. |
| (a) an object instance of a class, wherein: | [0032] - Page 10, lines 13-18. |
| (i) an initial value for one or more static properties of the class are assigned at run time; and | [0044] - Page 14, line 14- page 15, line 3 |
| (ii) the object instance has a dynamic property and an initial value of the dynamic property that are both generated and supplied, by a | [0010] - Page 4, line 22-page 5, line 7; [0034]-[0035] - Page 11, lines 5-21; FIG. 4 - 402-404; [0045] Page 15, lines 4-12; [0048]-[0049] - Page 16, line 13-page 17, line 8; FIG. 5 - 502; [0053] - Page 18, lines 3-12; |

| | |
|---|---|
| property source instance, at runtime for the object instance, on a per-instance basis and are not stored with the object; | |
| (b) an association between either: | [0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504; |
| (i) the object instance and the property source instance; or | [0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504; |
| (ii) the class and the property source instance; and | [0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504; |
| c) a user interface component that displays a collection of properties of the object instance including the one or more static properties and the dynamic property on a display device, wherein the user interface component is configured to: | [0010] - Page 4, line 22-page 5, line 7; [0034] Page 11, line 5-14; |
| (i) retrieve a reference to the object instance; | [0010] - Page 4, line 22-page 5, line 7; [0034]-[0035] - Page 11, lines 5-21; FIG. 4 - 402-404; [0045] Page 15, lines 4-12; [0048]-[0049] - |

| | |
|---|---|
| | Page 16, line 13-page 17, line 8; FIG. 5 - 502; [0053] - Page 18, lines 3-12; |
| (ii) retrieve the one or more static properties from the object instance; | [0044] - Page 14, line 14-page 15, line 3; FIG. 4, 410. |
| (iii) access the association to determine the property source instance associated with the object instance; | [0046] Page 15, lines 4-page 16, line 3; FIG. 4, 404-408; [0051] Page 17, lines 14-19; |
| (iv) call a method of the determined property source instance with the reference to the associated object instance; | [0034] Page 11, lines 5-14; [0061] - Page 20, line 19-page 21, line 7; FIG. 6, 602. |
| (v) receive the dynamic property, from the property source instance, wherein the property source instance dynamically generated the dynamic property and an initial value for the dynamic property; and | [0045] - Page 15, lines 4-12; FIG. 4, item 410; [0032]- page 10, lines 13-18; [0037]-[0038] - Page 12, lines 7-21; [0043] - Page 14, lines 4-13; FIG. 4 - 406 and 412; [0046] - Page 15, line 13-page 16, line 3; [0049] - Page 16, line 21-page 16, line 8; FIG. 5- steps 500-504; |
| (vi) display the static property and the dynamic property on the display device. | [0045] - Page 15, lines 4-12; FIG. 4, item 410; [0052]-[0053] - Page 17, line 20-page 18, line 12; FIG. 5, steps 512-520. |

Claims 9 and 13

These independent claims are directed towards the display of a property in a property list. More specifically, once an object with a property has been retrieved, that same object provides an ActiveX control that defines a user interface for displaying and editing the property. A list of properties is created. The ActiveX control is used to display a user interface for one of those

properties in the list. Thus, multiple ActiveX controls are used to display individual properties in a property list.

| CLAIM LIMITATION | SPECIFICATION SUPPORT |
|---|---|
| 9. A computer-implemented method for providing a custom graphical user interface for editing a property of an object, comprising: | [0058] - Page 19, line 20-page 20, line 3; FIG. 6; [0067]- Page 23, lines 5-11; FIG. 8. |
| receiving a first object having a first property, wherein the first object provides a custom ActiveX control that defines a first user interface for displaying and editing the first property; | [0067]- Page 23, lines 5-11; FIG. 8, step 802. |
| creating a list of one or more object properties to be displayed, wherein the list includes the first property; | [0068] - Page 23, lines 12-19; FIG. 8, step 804. |
| instantiating the custom ActiveX control; and | [0068] - Page 23, lines 12-19; FIG. 8, step 806. |
| displaying the object properties in the list, wherein the display of the first property comprises the first user interface defined by the instantiated custom ActiveX control, wherein the property may be edited through the first user interface. | [0069]- Page 23, line 20-page 24, line 2; FIG. 8, step 808. |
| | |
| 13. A system for providing a custom graphical user interface for editing a | [0058] - Page 19, line 20-page 20, line 3; FIG. 6; [0067]- Page 23, lines 5-11; FIG. 8. |

| | |
|---|---|
| property of an object comprising: | |
| (a) a computer; | [0022] - Page 7, lines 9-17; FIG. 1, 100. |
| (b) an application executing on the computer; | [0023] - Page 7, line 18-page 8, line 7; FIG. 1, 108; |
| (c) one or more objects, in the application, wherein each object has one or more object properties; | [0032]-[0033]- Page 10, line 13-page 11, line 2; Fig. 3, 300. |
| (d) a property inspector, in the application, configured to: | [0039] - Page 12, line 22-page 13, line 8; FIG. 4, 410. |
| (i) interrogate the one or more objects to discover one or more object properties to be displayed; | [0058]-[0059] - Page 19, line 20-page 20, line 11; FIG. 4, 410; FIG. 6. |
| (ii) create a list of the one or more object properties to be displayed; and | [0059] - Page 20, lines 4-11; FIG. 6 and FIG. 4, 410. |
| (iii) instantiate and host one or more property editors; | [0059] - Page 20, lines 4-11; FIG. 6 and FIG. 4, 410; [0060] - Page 20, lines 12-18; |
| (e) one or more property editors, in the application, wherein: | [0059] - Page 20, lines 4-11; FIG. 6 and FIG. 4, 410; [0060] - Page 20, lines 12-18. |
| (i) one of the property editors comprises a custom ActiveX control specified by one of the objects; and | [0061] - Page 20, line 19-page 21, line 7; FIG. 6, 602; FIG. 4, 410. |
| (ii) the custom ActiveX control defines a custom graphical user interface for displaying and editing one of the object properties. | [0059]-[0061] - Page 20, line 4-page 21, line 7; FIG. 6, 602. |

## VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-18 stand rejected under 35 U.S.C. §103(a) as being unpatentable over Hirsch (US 6,915,301)

These rejections are being appealed herein.

## VII. ARGUMENT

### A. Claims 1-18 are Patentable under 35 U.S.C. §103(a) over Hirsch

Independent claims 1, 5, 9, 13, and 17 were rejected as follows:

> Claim 1: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:
>
> a. receiving a reference to an object instance having a dynamic property that is created at runtime for the object instance on a per-instance basis and is not stored with the object (column 11, lines 37-58);
>
> Hirsch discloses retrieving reference to a property source instance (data source) from an association between the object and the property source instance (binding), wherein the property source instance creates and supplies to dynamic property (column 4, lines 11-34/column 11, lines 37-59), but does not explicitly disclose creating and supplying an initial value for the dynamic property for/to the object instance. However, Hirsch does disclose data sources generate values in a scene, and objects and their properties are bound to data sources in each scene. Furthermore, providing initial values was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.
>
> c. providing the reference to the object and the reference to the property source instance to a control which is configured to: (i) retrieve the dynamic property from the property source and (ii) display the property (object inspector window) in a user interface (column 11, 37-67).

> Claim 5: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:
>
> a. an object instance having a dynamic property that is created at runtime for the object instance on a per-instance basis and is not stored with the object (column 11, lines 37-59);
>
> Hirsch discloses a property source instance wherein the property source instance creates and supplies the dynamic property (column 11, lines 37-59) but does not explicitly disclose creating and supplying an initial value for the dynamic property for/to the object instance. However, Hirsch does disclose data sources generate values in a scene, and objects and their properties are bound to data sources in each scene (column 4, lines 11-34/column 11, lines 37-59). Furthermore, providing initial values was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create

and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.

c. an association between the object and the property source instance (column 3, lines 8-27);

d. a host configured to: (i) retrieve a reference to the object instance; (ii) retrieve a reference to the property source; (iii) provide the reference to the object and the reference to the property source to a control which is configured to (1) retrieve the dynamic property from the property source and (2) display the property in a user interface (column 11, lines 37-67).

Claim 9: Hirsch discloses a method and computer system for dynamic properties of software objects comprising receiving a first object having a first property wherein the first object provides a control (calendar control) that defines a first user interface for displaying and editing the first property, but does not explicitly disclose the control is an ActiveX control. However, Hirsch does disclose supporting ActiveX controls (column 12, lines 8-31). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to provide an ActiveX control to display and edit the first property. One would have been motivated to provide an ActiveX control to display and edit the first property in order to make the design more efficient.

b. creating a list of one or more object properties to be displayed, wherein the list includes the first property (column 12, lines 8-31);

c. instantiating the custom ActiveX control (column 12, lines 8-31);

d. displaying the object properties in the list, wherein the display of the first property comprises the first user interface defined by the instantiated ActiveX control, wherein the property may be edited through the first user interface (column 12, lines 8-31).

Claim 13: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:

a. one or more objects, wherein each object has one or more object properties (column 11, lines 37-59);

b. a property inspector configured to (i) interrogate the one or more objects to discover one or more object properties to be displayed; (ii) create a list of the one or more object properties to be displayed; (iii) instantiate and host one or more property editors (column 11, lines 37-59/column 12, lines 1-21);

Hirsch discloses one or more property editors wherein: (i) one of the property editors comprises a custom control specified by one of the objects, but does not explicitly disclose the control is an ActiveX control. However, Hirsch does disclose supporting ActiveX controls (column 12, lines 8-31). Therefore, it would have been obvious to one having ordinary skill in the art at the time the invention was made to provide an ActiveX control specified by one of the objects. One would have been motivated to provide an ActiveX control specified by one of the objects in order to make the design more efficient.

Hirsch discloses (ii) the custom ActiveX control defines a custom graphical user interface for displaying and editing one of the object properties (column 12, lines 8-31).

Claim 17: Hirsch discloses a method and computer system for dynamic properties of software objects comprising:

a. an object instance of a class, wherein (i) an initial value for one or more static properties of the class are assigned at run time; and (ii) the object instance has a dynamic property that is generated by a property source instance at runtime for the object instance on a per-instance basis and are not stored with the object (column 12, lines 20-58/column 11, lines 37-59). Hirsch does not explicitly disclose an initial value is generated and supplied by a property source instance. However, Hirsch does disclose data sources generate values in a scene, and objects and

> their properties are bound to data sources in each scene. Furthermore, providing initial values was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.
>
> b. an association between either: (i) the object instance and the property source instance; (ii) the class and the property source instance (column 11, lines 37-61)
>
> c. a user interface component that displays a collection of properties of the object instance including the one or more static properties and the dynamic property on a display device(column 11, lines 37-61), wherein the user interface component is configured to:
>
> (i) retrieve a reference to the object instance (column 11, lines 49-63);
>
> (ii) retrieve the one or more static properties from the object instance (column 11, lines 49-63);
>
> (iii) access the association to determine the property source instance associated with the object instance (column 11, lines 37-61);
>
> (iv) call a method of the determined property source instance with the reference to the associated object instance (column 11, lines 37-61);
>
> (v) receive the dynamic property, from the property source instance, wherein the property source instance dynamically generated the dynamic property (column 11, lines 49-63). Hirsch does not explicitly disclose an initial value is generated and supplied by a property source instance. However, Hirsch does disclose data sources generate values in a scene, and objects and their properties are bound to data sources in each scene. Furthermore, providing initial values was a well-known technique in the art at the time the invention was made. Therefore it would have been obvious to one having ordinary skill in the art at the time the invention was made that the data sources may supply an initial value for the dynamic property for/to the object instances in Hirsch. One would have been motivated to create and supply an initial value for the dynamic property in order to initialize the property to an initial value at runtime.
>
> (vi) display the static property and the dynamic property on the display device (column 11, lines 49-63).

Appellants traverse the above rejections. There are essentially two sets of claims. The first set includes independent claims 1, 5, and 17. The second set includes claims 9 and 13. Each set will be addressed separately herein.

1. Independent Claims 1, 5, and 17

The Office Actions rejects all of the claim limitations based on Hirsch. More specifically, Hirsch, Col. 11, lines 37-67 is used to reject the claims. Col. 11, lines 37-67 provides:

> Object properties consist of named attributes that define an object's appearance in terms of a functional expression. Access to these properties are provided through the Object Inspector. Their values are set at runtime based on the calculated result of the expressions. An expression may include the names of one or more data source columns which are automatically bound to a result row at runtime.

> Object property expressions result in one of the following base data types: Boolean, Numeric, String, Point, PointList, and Image. Derived Numeric types include Color, DateTime, Enum, Integer, and Percentage. Derived String types include FilePath (or URL) and FontName.
>
> Referring now to FIG. 2, the object inspector 30 is shown in more detail. The object inspector 30 provides two columns, an attribute name column 100 and a property column 110. The object being inspected in FIG. 2 has properties 112, 114, 116, 118, 120, 122, 124 and 126. Particularly, properties 114, 116, 118, 120, 124 and 126 are static constants. However, properties 112 and 122 are dynamic and are evaluated at run time. Due to the dynamic properties 112 and 122, to create a data driven application, the user only needs to enter the dynamic properties and code may be automatically generated. Thus, the data binding of the dynamic properties 112 and 122 is seamlessly integrated into the property sheet 30.
>
> The Object Inspector window 30 is a dockable control bar which displays an object's properties and events. The Object Inspector may be resized horizontally when docked and both vertically and horizontally when floating. The window may dock to the left or right of the world workspace when its location overlaps the docking site while being dragged.

As can be seen from this text, Hirsch completely and entirely fails to teach, describe, or suggest, explicitly or implicitly the creation by a property source instance of a dynamic property of a separate object instance. Hirsch further fails to teach, describe, or suggest the creation of a value for such a dynamic object. Appellants note that the term "dynamic" as set forth in the present claims refers to a property that is actually created and a value for that property is actually created dynamically. Hirsch does not utilize such a definition. Instead, Hirsch describes an object inspector that merely lists all of the properties of an object. Hirsch's "dynamic" property merely refers to properties 112 and 122 that are dynamic in the sense that they are evaluated at run time and are not static constants (such as properties 114, 116, 118, 120, 124, and 126). Thus, Hirsch's dynamic properties already exist for an object but are merely evaluated at run time. Such a teaching is not even remotely similar to the present claims which explicitly provide that a property source instance actually creates and supplies a dynamic property and a value for the dynamic property at run time while both the dynamic property and value are for a separate object instance.

The Office Action bypasses such a creation of the property and value elements and asserts that Hirsch does not explicitly disclose creating and supplying an initial value for a dynamic property to an object instance. Instead, the Action states that Hirsch discloses data sources that generate values in a scene, and objects and their properties are bound to data sources in each scene. However, once again, the claims are not directed towards merely generating values in a scene or binding objects and properties to data sources in a scene. Instead, the claims

explicitly provide for a property source instance creating and supplying both a dynamic property and an initial value for the dynamic property to a separate object instance. In addition, a control is used to retrieve the dynamic property from the property source instance and display the dynamic property. Nowhere in Hirsch is there even a remote reference to a control receiving both a reference to an object instance and a property source instance, and then retrieving the dynamic property from the property source instance and displaying such a property. Instead, Hirsch merely describes an object inspector window 30 that displays an object's properties and events. Hirsch fails to teach or allude to any control whatsoever or such a control receiving references to multiple object instances (i.e., a property source instance and an object instance) as claimed.

In addition to the above, the Office Action further relies on col. 4, lines 11-34 and col. 3, lines 8-27 to teach the various limitations of claim 5. Appellants note that col. 4, lines 11-34 are not relevant whatsoever to the present claim limitations. Such text merely describes the ability for a developer to build a virtual world having a scene containing information that is linked to data stored in a database. Hirsch builds data sources using a block diagramming tool which generates database queries. Hirsch's scenes are created with a drawing editor that binds data sources to the graphical elements of the scenes. However, what is wholly and completely lacking from such text is any description of a property source instance, an object instance, a property created by the property source instance for the object instance at run time, and/or a value for the property created by the property source instance for the object instance at run time (all of which are explicit claim limitations).

In addition, to teach the association claim limitation, the Office Action relies on col. 3, lines 8-27. Such text merely supports the arguments set forth above in that Hirsch's dynamic properties are merely properties that are evaluated at run time and are not static. Such a teaching does not and cannot teach the express claim limitations relating to the creating of both a dynamic property and a value for that property at run time.

In view of the above, Appellants submit that Hirsch does not and cannot teach, disclose, or suggest the invention as claimed.

In response to the above remarks, the final Office Action responds in paragraph (5). The response first asserts:

> Claims 1, 5, and 17: Applicant argues, "Hirsch [does not explicitly disclose] the creation of a property source instance for a property of a separate object instance." It is respectfully submitted that the disclosure of Claim 1 does not include this limitation.

Appellants respectfully disagree with and traverse such an assertion. As explained above and as explicitly claimed, a reference to an object instance is received. Further, a reference to a property source instance from an association between the object and the property source instance is retrieved. Further yet, the claims explicitly provide that the property source instance creates and supplies the dynamic property and an initial value for the dynamic property for/to the object instance. As can be clearly seen, the claims separately reference both the property source instance and the object instance. The claims also explicitly provide that an association between the two exists. Such claim language clearly provides for the creation by a property source instance of a property that is separate from the object instance. Further, the property source instance is clearly separate from the object instance. In this regard, to interpret the claim language differently lacks any foundation in both the claims and the specification.

The Action further continues and provides:

> Applicant argues, "Hirsch [does not explicitly disclose] the creation of a value for such a dynamic object." It is respectfully submitted that Hirsch creates and sets values at runtime based on calculated results of expressions. The creation of a value based on calculated expressions is viewed as dynamic because the value is created at runtime. Applicant argues Hirsch's dynamic properties already exist for an object but are merely evaluated at runtime, however, the creation of a value at runtime based on a calculated expression is dynamic.

Again, the claims not only provide for the dynamic creation of the value but for the dynamic creation of the property itself. The Action continues to assert that values are created at runtime. However, there is no mention of the dynamic creation of both the property and the initial value for the property. In this regard, Appellant's assertions above reflect that Hirsch's property already exists and the value for the property is evaluated at run time. Again, the claims provide for the dynamic creation of two (2) elements: (1) the property itself, and (2) the initial value for the property. Hirsch fails to even remotely allude to the dynamic creation of the property itself.

The Action further provides:

> Applicant argues, "Nowhere in Hirsch is there even a remote reference to a control receiving both a reference to an object instance and a property source instance, and then retrieving the dynamic property from the property source instance and displaying such a property." It is respectfully submitted that Hirsch discloses the limitation as claimed above. Specifically, Hirsch discloses an object inspector window displays the object, the object's properties (static and dynamic), and receives the object properties from a data source. The retrieval of this information by the object inspector window to display the information provides a reference between the property source and the object instances.

Appellants respectfully traverse such an assertion. As set forth in the Action itself, Hirsch's object inspector window displays the object, the object's properties, and receives the object properties from a data source. As can clearly be seen, Hirsch's data source is not an instance of an object or a property source instance as claimed. Again, the claims provide for a control that uses a reference to a property source instance to retrieve the dynamically created property. As claimed, the dynamic property is then displayed in the user interface. Hirsch, in col. 11, line 37-col. 12, line 30 describes a window that displays properties of an object with the left column containing the name of the property and the right column displaying the property's value. Values entered in the right column can be constants or can be calculated values containing functions or parameters or column names from a data source. Thus, rather than retrieving a dynamic property itself (as claimed), Hirsch merely describes the ability to retrieve a column name from a data source. Such a teaching is not even remotely similar to nor does it teach, disclose, suggest, allude to, or hint at retrieving a dynamic property itself (in addition to the dynamically created value for the property) and displaying such a property (as claimed).

Appellants question where Hirsch provides for dynamically creating both a property and a value for that property (as claimed). Appellants further question where Hirsch describes the ability to retrieve a dynamically created property from a property source instance and displaying the property.

In response to the above arguments, the Advisory Action first provides:

> Applicant disagrees with the assertion that Hirsch does not explicitly disclose the creation of a property source instance for a property of a separate object instance in the claimed limitations of Claim 1. The Examiner respectfully disagrees with the Applicant's traversal on the grounds that the claim language does not reflect the creation of a property source instance. The only creation step that is supplied in Claim 1 is the creation of a dynamic property, and an initial value, and not a property source instance.

Appellants respectfully disagree with and traverse such assertions. Firstly, Appellants do not assert that the claims provide for creating a property source instance. Such statements mischaracterize Appellant's arguments. Instead, Appellants asserted and continue to assert that the property source instance creates the dynamic property for the object instance and an initial value for such a property.

Examining claim 1, the first limitation (a) provides: "…an object instance having a dynamic property that is created at runtime…" Thus, as explicitly claimed, the limitation provides for creating a dynamic property at runtime. The next limitation (b) provides "…wherein the property source instance creates and supplies the dynamic property AND AN INITIAL VALUE FOR THE DYNAMIC PROPERTY for/to the object instance" (emphasis added). Thus, limitation (b) clearly provides for the creation of the initial value for the dynamic property (i.e., by the property source instance). Lastly, as stated above, the claimed object instance has a dynamic property which is created by the separate property source instance. In this regard, it is not the object instance that is creating the dynamic property and initial value, but instead, the property source instance creates the dynamic property and initial value for the object instance (see explicit claim language).

The Advisory Action continues:

> Applicant argues Hirsch's dynamic properties already exist for an object but are merely evaluated at runtime, however, the creation of a value at runtime based on a calculated expression is dynamic and is created on a per-instance basis.

Appellants note that evaluating a property at run time does not provide for creating the property that the value is assigned to. Again, the claims explicitly provide for creating both the property and the initial value for the property. Merely evaluating an expression for a property does not provide for nor does it create the property itself. Instead, as stated above, Hirsch's property already exists and a value for that property is merely evaluated at run time. Such a teaching neither teaches nor discloses, explicitly or implicitly the claimed property AND initial value for the property.

The Advisory Action continues:

> Applicant argues [Hirsch's] teaching does not teach, disclose, allude to, or hint at retrieving a dynamic property itself (in addition to the dynamically created value for the property) and displaying such a property (as claimed). It is respectfully submitted that Hirsch discloses the limitation as claimed above. Specifically, Hirsch discloses an object inspector window displays the object, the object's properties (static and dynamic), and receives the object properties from a data source. The retrieval of this information by the object inspector window to display the information provides a reference between the property source and the object instances.

Appellants respectfully disagree with and traverse the above assertions. Again, as described above, the claims provide for the creation of the property itself as well as the initial value for the property. All of the actions ignore the limitation relating to creating the property itself. Nowhere in Hirsch is there even a remote reference to the creation of such a property. Instead, the Advisory Action skips that portion of the claim, addresses the creation of the initial value, and proceeds to the last claim limitation relating to displaying the dynamic property in a user interface. Appellants reassert the arguments above with respect to the claimed "control". Namely, the control uses a reference to the property source instance to retrieve the dynamically created property. The Actions again merely ignore the "control" claim limitation and simply rely on an object inspector window that receives object properties from a data source (i.e., it merely receives the properties and displays them). However, as stated above, the claimed "control" receives a reference to the object instance and the reference to the property source instance. Hirsch's object window does not receive a reference to any such instances. Instead, data is retrieved from a data source. Hirsch's data source is not and cannot be equivalent to the claimed instances. Instead, Hirsch's data source merely has columns and column names are retrieved.

In response to the above arguments, the Examiner's Answer now asserts that:

> …the creation of a value for a property source instance is not supported by the claim language of claims 1, 5, and 17 based on the above interpretation.

Appellants respectfully disagree with and traverse the above assertions. Again, the claims do not provide for and the Appellants are not asserting the creation of a value for a property source instance. Instead, Appellants and the claims both provide that the property source instance creates both a dynamic property and an initial value for the dynamic property for the object instance. The Examiner is confused as to Appellants assertions and what is claimed. Again, as claimed, one object (referred to for argument purposes as Object 1) (i.e., the property

source instance) creates a property (i.e., the dynamic property) and a value for the property (i.e., the initial value) for a different object (referred to for argument purposes as Object 2). In other words, Object 1 creates a property for Object 2 and a value for that property.

Hirsch does not even remotely refer to such a teaching. Instead, as described above, Hirsch has objects with properties that already exist (and are not created by another object dynamically at runtime-as claimed) and values for such properties are evaluated and provided with values at runtime. Again, Hirsch is missing: (1) a separate object creating a property for another object; (2) the creation of both a property and a value for that property for the separate object; and (3) the dynamic creation of both the property and the value.

The Answer then asserts that data sources are named SQL queries that are executed and used in a layout. The Answer continues and provides that when used in a layout, each row resulting from a query is transformed into a graphical representation represented by a data element that consists of objects whose properties contain values that are set at runtime based on calculated expressions, making them dynamic properties. The Answer further asserts that such a teaching leads to the conclusion that dynamic values are created for dynamic properties. Appellants again respectfully traverse such an assertion. Again, properties are not being created whatsoever in Hirsch. Such property creation is explicitly and expressly required in the claims. Thus, the Examiner is attempting to ignore an explicit claim limitation relating to the "creation" aspect and instead focuses on a definition of a "dynamic property". The claims do not provide that the property is dynamic since it has dynamic values for the properties. Instead, as asserted numerous times over the prosecution history, the express claim language provides for dynamically creating the property itself by one object and also dynamically creating a value for that property. Hirsch's disclosure of executing queries to retrieve already created properties does not reflect the creation of a property by one object for a second object, nor does it reflect the creation of a property whatsoever.

With respect to the "control" related elements of the claims, Appellants reassert the above arguments. The Answer asserts that Hirsch's object inspector window displays the object, the objects properties, and receives the object properties from the data source. The Answer further asserts that the retrieval of the information by the object inspector window provides the teaching

of a reference between the property source (data source) and the object instances (objects). Appellants note that the Examiner is again misinterpreting the claim language. The claims explicitly provide for providing two references: (1) a reference to the object instance, and (2) a reference to the property source instance. The control related aspects of the claims do not provide for nor require a reference between the property source and object instances. Instead, there are two explicitly claimed references that are provided and used. The Office Actions and Answer fail to acknowledge such claim limitations and continue to misapply the reference to the present claims.

In view of the above, Appellants respectfully submit that the failure to consider explicitly claimed limitations amounts to an improper rejection, a rejection that fails to establish a prima facie case of unpatentability, and a rejection that is clearly in error. Accordingly, Appellants respectfully disagree with and traverse the Examiner's rejections and respectfully request that the rejections be reversed and the claims allowed.

2. Dependent Claims 2 and 6

Dependent claims 2 and 6 provide that the dynamic property is provided by an application that is extending an object property set of the object.

In rejecting these claims, the final Office Action merely refers to Hirsch col. 12, lines 41-58. Such text describes Hirsch FIG. 3 as an object model having a base class (referred to as VcPropertyBag 200) that acts as a container for a set of properties specific to a derived class. The base class allows properties to be accessed and manipulated.

However, what is clearly lacking from any such description is a reference to an application (as claimed) whatsoever. Instead, an object model is described along with classes that may be derived therefrom. Such an object model does not remotely reflect or relate to an application that is extending an object property set of an object. Further, when reading these claim limitations in combination with the independent limitations, the dynamic property is provided by the application at run time on a per-instance basis. Nowhere is there even a remote hint of such a practice or implementation in Hirsch.

In response to the above, the Answer asserts that Hirsch discloses a virtual world associated with an application that is built using building blocks such as scenes, data sources, global parameters and resources; that data sources provide objects through the creation of data elements; that an objects property set includes named attributes that define an object's appearance in terms of a functional expression; that values are set at runtime based on the calculated result of the expressions; and that the property set is extended through the use of calculated expressions in the sense that different expressions provide various calculated results.

Appellants respectfully disagree with and traverse such assertions. Hirsch does not extend a property set or properties for an object whatsoever. Instead, Hirsch evaluates values for already existing properties using expressions. There is a clear difference between extending a property set (as claimed) versus evaluating values for properties (as in Hirsch). Further, nowhere in Hirsch is there a reference to an application that extends such a property set, either at runtime, or on a per-instance basis.

In view of the above, Appellants respectfully request reversal of the Examiner's rejections of these claims.

3. Dependent Claims 3 and 7

These dependent claims provide that the reference to the property source instance is retrieved from a mapping of property source instances to object class. In other words, these claims provide that the claimed association in the independent claims consists of a mapping.

In rejecting these claims, the Office Action relies on Hirsch col. 3, lines 8-27 and column 12, lines 41-58. Col. 3, lines 8-27 merely is part of Hirsch's summary of invention and describes the advantages of Hirsch's invention with respect to dynamic object properties over static object properties and data binding. However, nowhere in the text is there even a remote hint at a mapping between property source instances and object class. The rejection equates the claimed mapping to the describe binding. Hirsch's described binding relates to binding data to an entire class of object:

> Furthermore, the bindings are applied to an entire class of object (all objects represented by the data element node) rather than simply to a single object (i.e., the calculated value for each object is dependent upon the row represented by the object).

Thus, such text binds a class of an object to a column or row of a data source. Such a teaching does not relate to, explicitly or implicitly, the ability to map a property source instance to an object class and retrieving a reference to a property source instance from such a mapping as claimed. Again, Hirsch's data source is not equivalent to the claimed property source instance nor is it equivalent to the claimed object class. Instead, it is merely a data source with rows and columns. The claims are explicit and specific and the cited references fail to teach such limitations.

The text in col. 12, lines 41-58 again merely describes the base classes and derived classes (see above). Such text fails to teach and is not even remotely relevant to the claims limitations.

In response to the above, the Answer essentially asserts the same arguments. Again, the claims here provide for retrieving the reference to the property source instance from a mapping of property source instances to object class and NOT a binding of an object to a column or row of a data source as set forth in Hirsch.

In view of the above, Appellants respectfully request reversal of the rejections.

4. Dependent Claims 4 and 8

These dependent claims provide for retrieving the standard properties of the object (i.e., the properties that are not dynamically created by the property source instance) and displaying such standard properties. Such claims serve to distinguish the different types of properties of the object.

In rejecting these claims, the Office Action relies on Hirsch col. 11, lines 52-67 which merely describes the object inspector displaying multiple columns having the dynamic and static properties displayed. Thus, both the Office action and Hirsch rely on the same object inspector display to display the properties. Again, the claims provide for different types of properties - standard properties and the dynamically created properties - the Action fails to acknowledge such differences and Hirsch fails to teach such differences.

In response to the above arguments, the Answer merely repeats the same assertions. Appellants respectfully disagree and note that the different types of objects as provided in these dependent claims are nowhere to be found in Hirsch.

In view of the above, Appellants respectfully request reversal of the rejections.

5. Independent Claims 9 and 13

These independent claims are directed towards the display of a property in a property list. More specifically, once an object with a property has been retrieved, that same object provides an ActiveX control that defines a user interface for displaying and editing the property. A list of properties is created. The ActiveX control is used to display a user interface for one of those properties in the list. Thus, multiple ActiveX controls are used to display individual properties in a property list.

Appellants submit that Hirsch clearly fails to teach, disclose, or suggest such unique attributes as set forth in the present claim limitations. In rejecting these claims, the office Action asserts that Hirsch's calendar control teaches the invention as claimed (col. 12, lines 8-31). Col. 12, lines 8-31 describes a 2-column entry form for setting the values of a selected object's properties. The left column displays the name of the property while the right column is a read/write column that displays the property's value. The values entered can be constants or can be calculated containing functions or parameters or column names from a data source. If a property value is of an enumerated type, a drop down combo box may be displayed listing the legal values. Further, if the property is a date or time, a calendar control may be displayed beneath the property value when the field is active.

However, what should be noted is that nowhere in Hirsch is there any description that the object itself provides the custom control to display the properties of the object. Instead, Hirsch explicitly teaches a defined set of controls that are used independent of the object. In this regard, Hirsch's object does not provide the combo box that is used to display a list of enumerated properties. Instead, an object inspector provides a list of properties and determines what controls are used to edit the properties. Such controls are thus provided by the object inspector and not the object whose properties are being displayed in the tabbed sheet (as claimed). In this regard,

the object inspector's properties are not being displayed in the tabbed sheet. Instead, the object inspector is used to display a different object's properties.

In contrast to Hirsch's teaching, the presently claimed invention provides that the object provides a custom control that defines a user interface for displaying and editing one of its own properties. Such a capability is wholly and completely missing, both explicitly and implicitly from Hirsch.

Further, Hirsch does not even remotely describe that the control is a custom control. Instead, Hirsch's controls are statically defined based on the type of property being displayed (e.g., if the property value is of an enumerated type, a drop-down combo box is used or if it is a date or time, a calendar control is used). Such a control is not a custom control provided by the object and used to edit a particular property of the object at the choice of the object itself.

In response to the above arguments, the Office Action first provides:

> Claims 9 and 1 3: Applicant argues, "Nowhere in Hirsch is there any description that the object itself provides the custom control to display the properties of the object." It is respectfully submitted that Hirsch discloses the limitation as claimed above. The objects in Hirsch are selected in an object inspector window, and provide custom controls such as calendar controls when specific objects are inspected. An object that does not contain a date or time property would not provide a calendar control, giving the objects the ability to provide custom controls based on which object is being inspected.

Appellants reassert that arguments set forth above. Namely, the present claims provide the ability for the object itself to provide a custom control to display one of its own properties. The Action merely asserts that the object inspector window provides controls to display another object's properties. Such a teaching is not what the present claims teach, disclose, nor suggest and cannot and does not render the present claims obvious. Again, Hirsch's displaying of another object's properties via a predefined set of controls is an entirely different implementation that the claimed objects displaying their own set of properties. The Action merely ignores such a distinction and provides that an object that does not contain a date or time property would not provide a calendar control. The relevance of such a statement to the present claims is lacking and ignores the differences set forth above.

The Office Action continues and provides:

> Applicant argues, "Hirsch does not even remotely describe that the control is a custom control." It is respectfully submitted that the controls provided in Hirsch are custom controls due to the fact that different controls are used for editing particular properties of an object.

Appellants respectfully disagree with and traverse such a rejection. Again, the present invention provides not only that the control is provided by the object itself, but the control is used to display and edit the property of the object itself in a list of properties, and the control is a custom control that defines a user interface for displaying and editing the property. As described above, Hirsch fails to provide such a custom control that defines a user interface for displaying and editing the property. Instead, Hirsch merely selects a control to use from an enumerated list of controls. There is not even a remote hint at defining a custom control. Further, the claimed custom control is used to display the property from the object itself and is not in the form of Hirsch's object inspector that is displaying a property from another object.

In view of the above, Appellants assert that the various elements of Appellants' claimed invention together provide operational advantages over the systems disclosed in Hirsch. In addition, Appellants' invention solves problems not recognized by Hirsch.

In response to the above previously submitted arguments, the Advisory Action provides:

> Applicant argues "Nowhere in Hirsch is there any description that the object itself provides the custom control to display the properties of the object." It is respectfully submitted that Hirsch discloses the limitation as claimed above. The objects in Hirsch are selected in an object inspector window, and with broad, reasonable interpretation, "provide" custom controls such as calendar controls when specific objects are inspected. An object that does not contain a date or time property would not provide a calendar control, giving the objects the ability to provide custom controls based on which object is being inspected. Applicant argues, "Hirsch does not even remotely describe that the control is a custom control." It is respectfully submitted that the controls provided in Hirsch are custom controls due to the fact that different controls are used for editing particular properties of an object

As can be seen by the above assertions, the Advisory Action fails to address the differences noted above and merely reasserts the previous rejections/assertions. Appellants respectfully repeat the arguments set forth above. In addition, Appellants note that the ability for an object to provide its own control to edit one of its own properties provides significant advantages over the use of standard controls offered by an entirely different object (i.e., Hirsch's object inspector). For example, the object itself need not rely on other objects. Such advantages are lacking in Hirsch and not acknowledged nor addressed in the various Patent Office Actions.

In response to the above arguments, the Answer again asserts that Hirsch's object inspector window displays object properties. As stated above, such a teaching is entirely and completely different from that set forth and explicitly claimed.

The Answer then asserts that based on the broadest, reasonable interpretation, the objects provide these custom controls. Appellants respectfully disagree with and traverse such an assertion. Hirsch is not unclear nor ambiguous in what it describes. As described in Hirsch and detailed herein, Hirsch's object inspector window is used to display properties of another object. The present claims explicitly provide "the first object provides a custom ActiveX control that defines a first user interface for displaying and editing the first property". Thus, the claims expressly require that the object itself provides the ActiveX control. A different object such as an object inspector window is not and cannot fall within the scope of such claim language.

With respect to the claimed "custom" aspects of the "custom control", the Answer asserts that since a user can enter a value rather than using a calendar control, the control is "custom" as claimed. Appellants respectfully disagree with and traverse such assertions. The claims provide for a custom control that is provided by the object itself, and that serves to define a user interface for displaying and editing a property of the object. The ability to enter a value for a date is not a control that defines a user interface for displaying and editing a property. In this regard, similar to the arguments above, the Examiner is attempting to compare elements of Hirsch that simply do not match up to the present claims. In this regard, the ability to use a calendar control or to not use a calendar control to enter a value for a property simply does not even remotely allude to a custom ActiveX control that defines a user interface for displaying and editing a property for the object itself (as claimed).

In view of the above, Appellants respectfully request reversal of the rejections.

6. Dependent Claims 10 and 14

These dependent claims provide for instantiating stock ActiveX controls that define additional user interfaces for displaying and editing remaining object properties in the list (i.e., thereby distinguishing stock ActiveX controls from custom ActiveX controls). Further, the

claims explicitly provide that the stock ActiveX controls are not provided by any object containing one or more of the remaining object properties.

The last part of this claim limitation clearly differentiates Hirsch from the present invention. In this regard, if one accepts the assertion in the Office Action that the object inspector is equivalent to the claimed controls, then based on these claims, the object inspector cannot provide any control to display other properties. The Action simply relies on the object inspector for all of the properties. However, it is impossible for the object inspector to teach both the stock ActiveX controls and the custom ActiveX controls since this claim provides that the stock ActiveX controls are not provided by the object containing the remaining object properties. Based on these claims, Hirsch's object inspector could not provide both types of ActiveX controls. However, the Action relies on Hirsch's object inspector for just that purpose. Accordingly, the Action fails to acknowledge the claim limitations and is thus clearly in error.

In response to the above arguments, the Answer provides that the object inspector provides a 2-column entry form and a user may use a drop-down combo box (stock control) or a calendar control (custom control) to edit the value of an object. The Answer further asserts that the controls are not provided by any object containing one or more of the remaining object properties in the sense that a object that only contains date and time properties would not provide the stock combobox controls.

Appellants respectfully traverse such assertions. Firstly, such an analysis is illogical. If the objects that only have date and time properties do not provide the stock combobox controls, then the claim limitation of providing stock ActiveX controls has not been met. Thus, under the Examiner's analysis, the claim limitations have not been taught by Hirsch. Further, as described above, stock controls are provided by an object other than the object containing the properties while the custom controls are provided by the object itself. However, Hirsch's object inspector provides all of the controls without exception. Thus, the Answer's arguments are both illogical and inconsistent.

In view of the above, Appellants respectfully request reversal of the rejections.

7. Dependent Claims 11 and 15 Are Not Separately Argued

8. Dependent Claims 12 and 16

These dependent claims provide for an API that utilizes a push model wherein a first object is pushed to a second object for display.

In rejecting these claims, the Office Action merely relies on Hirsch col. 12, lines 41-58 which describes an object model and base and derived classes. However, the use of a push based model or the ability to push a first object to a second object for display is not even remotely hinted at anywhere in the cited text or remainder of Hirsch.

In response to the above arguments, the Answer asserts that objects are pushed to the object inspector window in order to examine and edit their properties. Appellants assert that such an assertion is meritless. In this regard, Hirsch's object inspector provides access to certain properties and their values. However, nowhere in Hirsch is there any capability to push (i.e., a push based system) where an object is pushed to the object inspector for display. Instead, the object inspector must request and retrieve such values.

In view of the above, Appellants respectfully request reversal of the rejections.

9. Dependent Claim 18

Dependent claim 18 provides that the custom ActiveX controls are provided on a per-property basis. The claim further provides that two or more object properties are in the properties list and are displayed in a list. Each of the two or more properties are displayed using interfaces defined by the custom ActiveX controls.

In rejecting these claims, the Office Action asserts that in view of Hirsch, it would be obvious to provide an ActiveX control on a per-property basis in order to make the design more efficient. Appellants respectfully disagree. In this regard, it would appear that to be more efficient, a single ActiveX control used to display all properties would be more efficient than using an ActiveX control on a per-property basis. Thus, the Examiner's reliance upon motivation would actually serve to teach away from the present invention. Further, Hirsch fails to even remotely describe using custom ActiveX controls on a per-property basis as claimed.

Again, the claims provide for using custom ActiveX control on a per property basis to display the properties in a list via user interfaces defined by the custom ActiveX controls. Nowhere in Hirsch has such a teaching been contemplated, suggested, or described, explicitly or implicitly.

In response to such arguments, the Answer first asserts that by providing controls on a per property basis, operator efficiency would be increased. The Answer provides an example that a calendar control enables a user to quickly enter a date as opposed to entering the day, month, and year individually. Appellants note that the Examiner is missing the point of Appellants' argument. In this regard, Appellants argue not that a custom control is less efficient, but that Hirsch's disclosure provides for using only the object inspector and not for multiple different custom controls. In this regard, Appellants note that for efficiency with respect to computer programming, it may be more efficient to use a single control like an object inspector window rather than a different custom control for each and every property of an object.

The answer further asserts that different controls are provided on a per-property basis and includes an example of a drop down combo-box for one property and a calendar control for another property. Such an argument is wholly and completely inconsistent with the prior assertions by the Examiner. In this regard, the Examiner previously asserted that the drop down combo-box was not a custom control but was instead a stock control. The present claims provide for multiple different custom controls. Thus, Hirsch's stock control cannot be used to teach the presently claimed custom control. Such an assertion in the answer is therefore meritless, misleading, inconsistent, in error, and fails to establish a prima facie case of unpatentability.

In view of the above, Appellants respectfully request reversal of the rejections.

VIII. Conclusion

In light of the above arguments, Appellants respectfully submit that the cited references do not anticipate nor render obvious the claimed invention. More specifically, Appellants' claims recite novel physical features which patentably distinguish over any and all references under 35 U.S.C. §§ 102 and 103. As a result, a decision by the Board of Patent Appeals and Interferences reversing the Examiner and directing allowance of the pending claims in the subject application is respectfully solicited.

Respectfully submitted,

GATES & COOPER LLP

Attorneys for Appellant(s)

Howard Hughes Center
6701 Center Drive West, Suite 1050
Los Angeles, California 90045
(310) 641-8797

Date: November 18, 2008

By: /Jason S. Feldmar/
Name: Jason S. Feldmar
Reg. No.: 39,187

JSF/

G&C 30566.296-US-U1

## CLAIMS APPENDIX

1. (PREVIOUSLY PRESENTED) A computer-implemented method for displaying per-instance dynamic properties of an object comprising:

(a) receiving a reference to an object instance having a dynamic property that is created at runtime for the object instance on a per-instance basis and is not stored with the object;

(b) retrieving a reference to a property source instance from an association between the object and the property source instance, wherein the property source instance creates and supplies the dynamic property and an initial value for the dynamic property for/to the object instance; and

(c) providing the reference to the object instance and the reference to the property source instance to a control, wherein the control is configured to:

(i) retrieve the dynamic property from the property source instance; and

(ii) display the dynamic property in a user interface.

2. (ORIGINAL) The method of claim 1, wherein the dynamic property is provided by an application that is extending an object property set of the object.

3. (ORIGINAL) The method of claim 1, wherein the reference to the property source instance is retrieved from a mapping of property source instances to object class.

4. (ORIGINAL) The method of claim 1, wherein the control is further configured to:

retrieve the standard properties for the object; and

display the standard properties.

5. (PREVIOUSLY PRESENTED) A computer-implemented system for displaying per-instance dynamic properties of an object comprising:

(a) a computer;

(b) an application executing on the computer;

(c) an object instance, in the application, having a dynamic property that is created at runtime for the object instance on a per-instance bases and is not stored with the object;

(d) a property source instance, in the application, wherein the property source instance creates and supplies the dynamic property and an initial value for the dynamic property for/to the object instance;

(e) an association, in the application, between the object and the property source instance; and

(f) a host, in the application, configured to:

(i) retrieve a reference to the object instance;

(ii) retrieve a reference to the property source instance from the association; and

(iii) provide the reference to the object instance and the reference to the property source instance to a control, wherein the control is configured to:

(1) retrieve the dynamic property from the property source instance; and

(2) display the dynamic property in a user interface.

6. (ORIGINAL) The system of claim 5, wherein the dynamic property is provided by an application that is extending an object property set of the object.

7. (ORIGINAL) The system of claim 5, further comprising a mapping of property source instances to object classes, wherein the host is configured to retrieve the reference to the property source instance from the mapping.

8. (ORIGINAL) The system of claim 5, wherein the control is further configured to:
retrieve the standard properties for the object; and
display the standard properties.

9. (ORIGINAL) A computer-implemented method for providing a custom graphical user interface for editing a property of an object, comprising:

receiving a first object having a first property, wherein the first object provides a custom ActiveX control that defines a first user interface for displaying and editing the first property;

creating a list of one or more object properties to be displayed, wherein the list includes the first property;

instantiating the custom ActiveX control; and

displaying the object properties in the list, wherein the display of the first property comprises the first user interface defined by the instantiated custom ActiveX control, wherein the property may be edited through the first user interface.

10. (ORIGINAL) The method of claim 9, further comprising instantiating one or more stock ActiveX controls that define one or more additional user interfaces for displaying and editing remaining object properties in the list, wherein the stock ActiveX controls are not provided by any object containing one or more of the remaining object properties.

11. (ORIGINAL) The method of claim 10, wherein the first user interface and additional user interfaces are displayed in a single dialog box.

12. (ORIGINAL) The method of claim 9, wherein an application programming interface provides the ability to push the first object to a second object for display.

13. (PREVIOUSLY PRESENTED) A system for providing a custom graphical user interface for editing a property of an object comprising:

(a) a computer;

(b) an application executing on the computer;

(c) one or more objects, in the application, wherein each object has one or more object properties;

(d) a property inspector, in the application, configured to:

(i) interrogate the one or more objects to discover one or more object properties to be displayed;

(ii) create a list of the one or more object properties to be displayed; and

(iii) instantiate and host one or more property editors;

(e) one or more property editors, in the application, wherein:

(i) one of the property editors comprises a custom ActiveX control specified by one of the objects; and

(ii) the custom ActiveX control defines a custom graphical user interface for displaying and editing one of the object properties.

14. (ORIGINAL) The system of claim 13, wherein one the property editors is comprised of a stock ActiveX control that defines an additional user interface for displaying and editing one or more additional properties in the list, wherein the stock ActiveX control is not provided by one of the objects that contains the one or more additional properties.

15. (ORIGINAL) The system of claim 14, wherein the custom graphical user interface and additional user interfaces are displayed in a single dialog box.

16. (ORIGINAL) The system of claim 13, further comprising an application programming interface configured to provide the ability to push the one or more object to the property inspector for display.

17. (PREVIOUSLY PRESENTED) A computer-implemented system for displaying per-instance dynamic properties of an object comprising:

(a) an object instance of a class, wherein:

(i) an initial value for one or more static properties of the class are assigned at run time; and

(ii) the object instance has a dynamic property and an initial value of the dynamic property that are both generated and supplied, by a property source instance, at runtime for the object instance, on a per-instance basis and are not stored with the object;

(b) an association between either:

(i) the object instance and the property source instance; or

(ii) the class and the property source instance; and

(c) a user interface component that displays a collection of properties of the object instance including the one or more static properties and the dynamic property on a display device, wherein the user interface component is configured to:

(i) retrieve a reference to the object instance;

(ii) retrieve the one or more static properties from the object instance;

(iii) access the association to determine the property source instance associated with the object instance;

(iv) call a method of the determined property source instance with the reference to the associated object instance;

(v) receive the dynamic property, from the property source instance, wherein the property source instance dynamically generated the dynamic property and an initial value for the dynamic property; and

(vi) display the static property and the dynamic property on the display device.

18. (PREVIOUSLY PRESENTED) The method of claim 9, wherein:

custom ActiveX controls are provided on a per-property basis;

two or more object properties are in the properties list; and

the two or more object properties are displayed in a list, wherein each of the two or more object properties are displayed using user interfaces defined by the custom ActiveX controls.

## EVIDENCE APPENDIX

None.

## RELATED PROCEEDINGS APPENDIX

None.